RT-Lab Solo

Getting Started User's Manual

IA Lab

Center for Intelligent Machine
McGill University

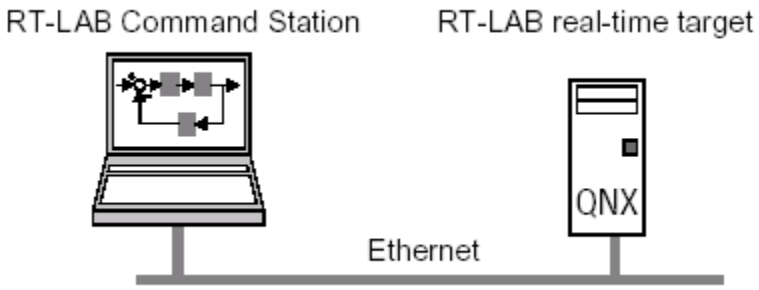
Version: 2.11

Mar. 27, 03

1 Fundamental

1.1 How RT-LAB Solo Works

RT-LAB Solo runs on a hardware configuration consisting of Command Station, Compilation Node, Target Node, Communication Links (real-time and Ethernet), and I/O boards.



1.1.1 Command Station

RT-LAB software is configured on a Windows NT or Windows 2000 computer called the Command Station. The Command Station serves as the user interface. It allows users to:

• Edit and modify models;
• See model data;
• Run the original model under its simulation software (Simulink etc.);
• Generate code;
• Separate code;
• Control the simulator's Go/Stop sequences.

1.1.2 Target Node

Target Node is the computer where simulation runs. For real-time simulation, the preferred operating system for the Target Node is QNX. When there are multiple QNX nodes, one of them is assigned as the Compilation Node. The Command Station and Target Node(s) communicate with each other using communication links, and for hardware-in-the-loop simulations Target Node may also communicate with other devices through I/O boards.

The real-time Target Node perform:

• Real-time execution of the model's simulation;
• Real-time communication between the nodes and I/Os;
• Initialization of the I/O systems;
• Acquisition of the model's internal variables and external outputs through I/O modules;
• Implementation of user-performed online parameters modification;

• Recording data on local hard drive, if desired;
• Supervision of execution of the model's simulation, and communication with other Nodes.

1.1.3 Compilation Node

The Compilation Node is used to:
• Compile C code;
• Load the code onto each Target Node;
• Debug the user's source code (S-function, User Code Block, etc.).

For RT-Lab Solo, Command Station and Compilation Node are referred to the same computer.

1.2 How RT-LAB is Used

RT-LAB is an industrial-grade software package for engineers who use mathematical block diagrams for simulation, control, and related applications. The software is layered on top of industry-proven commercial-off-the-shelf (COTS) components like popular diagramming tools MATLAB/Simulink and works with viewers such as LabVIEW and programming languages including Visual Basic and C++.

1.2.1 Designing and validating the model

The starting point for any simulation is a mathematical model of the system components that are to be simulated. RT-LAB is designed to automate the execution of simulations for models made with offline dynamic simulation software, like Simulink, in a real-time environment. RT-LAB is fully scalable, allowing users to separate mathematical models into blocks to be run in parallel on a cluster of machines, without subtly changing the model's behavior, introducing real-time glitches, or causing deadlocks. The detailed steps of accommodating a Simulink model into RT-Lab will be discussed in the following chapters.

1.2.2 Using block diagrams

Using block diagrams for programming simplifies the entry of parameters, and guarantees complete and exact documentation of the system being modeled. Once the model is validated, the user separates it into subsystems and inserts appropriate communication blocks. Each subsystem will be executed by Target Node in RT-LAB's distributed system. The detailed steps of grouping and adding communication blocks will be discussed in the following chapters.

1.2.3 Running Simulations

Once the original model has been separated into subsystems associated with the various

processors, each portion of the model is automatically coded in C, and compiled for execution by the Target Node.

Target Node is commercial PCs, equipped with PC-compatible processors, which operate under a QNX environment. In the QNX environment, the real-time sending and reception of data between QNX nodes is performed through FireWire-type communication boards, typically at 200 Mb/s or 400 Mb/s (depending on the card chosen).

More detailed description of Running Your Simulation can be found in the following chapters.

1.2.3.1 RT-LAB simulation

When the C coding and compilation are complete, RT-LAB automatically distributes its calculations among the Target Node, and provides an interface so users can execute the simulation and manipulate the model's parameters. The result is high-performance simulation that can run in parallel and in real-time.

1.2.3.2 Using the Console as a graphic interface

Users can interact with RT-LAB during a simulation by using the Console, a command terminal operating under Windows NT. Communication between the Console and the Target Node is performed through a TCP/IP connection. This allows users to save any signal from the model, for viewing or for offline analysis. It is also possible to use the Console to modify the model's parameters while the simulation is running.

2 Building a Distributed Model for RT-LAB

Any Simulink model can be implemented in RT-LAB, but some modifications must be made in order to distribute the model and transfer it into the simulation environment. The success of distributed computing with a complex model will depend on the separation of that model into small subsystems synchronized to run in parallel. This should be kept in mind early in and throughout the design process. To build an RT-LAB model, users must modify the block diagram (.mdl file) by:

- Regrouping the model into calculation subsystems
- Inserting *OpComm* communication blocks

Each of these topics is covered within this chapter. After these steps have been completed, RT-LAB begins the compilation process with the regrouped file, separating the model, and generating and compiling the code. The user then sets execution settings, which are covered in the following chapters, after which point the model's simulation is ready to be executed.

This chapter explains the steps required to build an RT-LAB model by modifying your Simulink block diagram to ensure it is properly separated for distributed execution, and maximize the performance:

1. Group the model into subsystems.
In this step, you graphically group into subsystems the calculations to be performed by a given CPU.

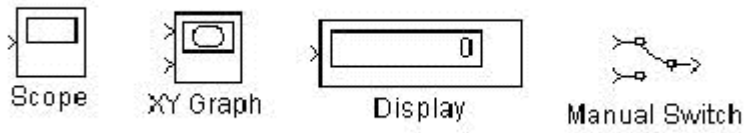
2. Insert *OpComm* communication blocks.
Communication blocks are part of a block library specifically defined for the RT-LAB interface. They are used by RT-LAB to identify parameters required for communication between the nodes in the hardware configuration.

2.1 Building RT-Lab Model

The model must be divided into subsystems, each of which represents one node in the real-time network. Two types of subsystems are available:

Console:

The Console subsystem is the station operating under Windows NT, where the user interacts with the system. It contains all the Simulink blocks related to acquiring and viewing data (scope, manual switch, etc.). Any of these blocks which are required by the user, whether during or after the execution of the real-time model, should be included in the *Console Subsystem*. There can be only one Console per model.



Some Simulink blocks that may be included in the *Console Subsystem*.

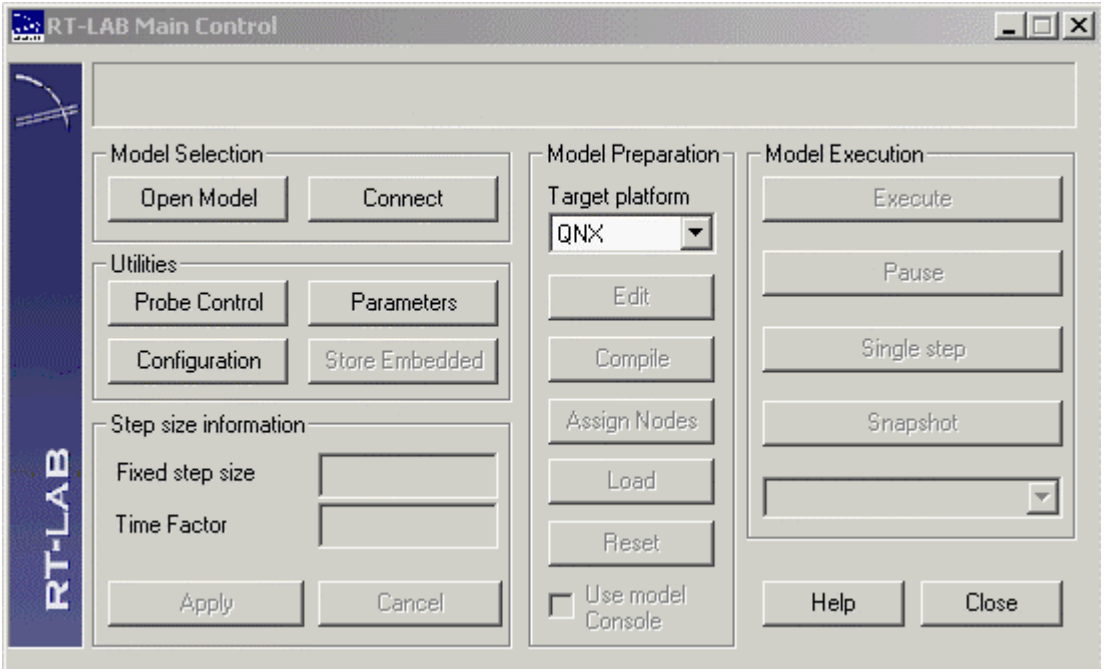
Master:

The Master computation subsystem is responsible for the model's real-time calculation and for the overall synchronization of the network. In a system containing Hardware-in-the-Loop (HIL) this subsystem is also responsible for I/O communication. The Master includes Simulink blocks that represent operations to be performed on signals or on I/O icons. There can be only one Master subsystem per model.

A demonstration model is supplied with the system. This model has been designed to function properly in the RT-LAB environment, and allows you to review all the steps required to operate the system. Using the information provided in this Manual, you should be able to successfully open the demonstration file, group the model into its required subsystems, set its parameters, and run its simulation on your cluster.

2.1.1 Start RT-Lab System

You can find the RT-Lab's shortcut on the desktop of Copper. The name of the icon is "MainControl" Double click on it and the system will be started. The following figure is the Main Control panel.



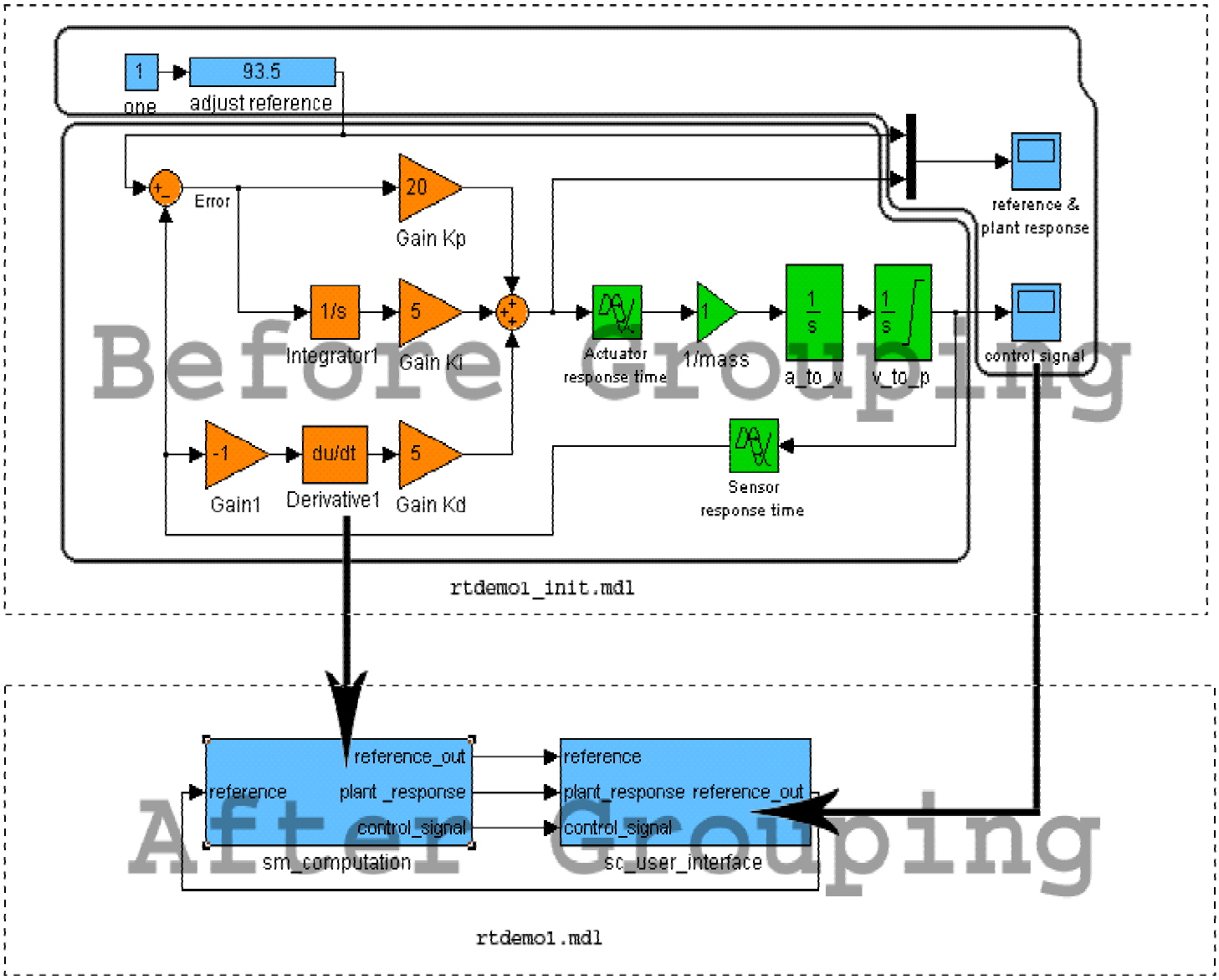
2.1.2 Open the Demonstration file

Click on the button "Open Model". This enables you the selection of the Simulink file for the model to be processed and executed. The demo file is in the directory C:/ Opal-RT / RT-Lab/ Simulink / models / rtdemo1.mdl.

2.1.3 Group the Subsystems

The model in the file rtdemo1.mdl has already been properly grouped into subsystem, and is ready to be executed by RT-Lab. The original system can be found in the same directory with the name rtdemo1_init.mdl

The following Diagram shows the process of grouping this demo model.



2.1.4 Rename the Subsystem

For any given model, there must be only one master calculation subsystem, and its name must begin with the prefix *SM_*. There is also only one *Console Subsystem*, and its name must begin with the prefix *SC_*.

As you may have seen in the figure of the previous page, the two subsystems have been renamed to *SM_Computation* and *SC_User_Interface*. The first one is the *Master Computation Subsystem* and the second one is the *Console Subsystem.*

2.1.4.1 To keep in mind when dividing your model

A number of conventions must be followed when organizing and naming the subsystems.

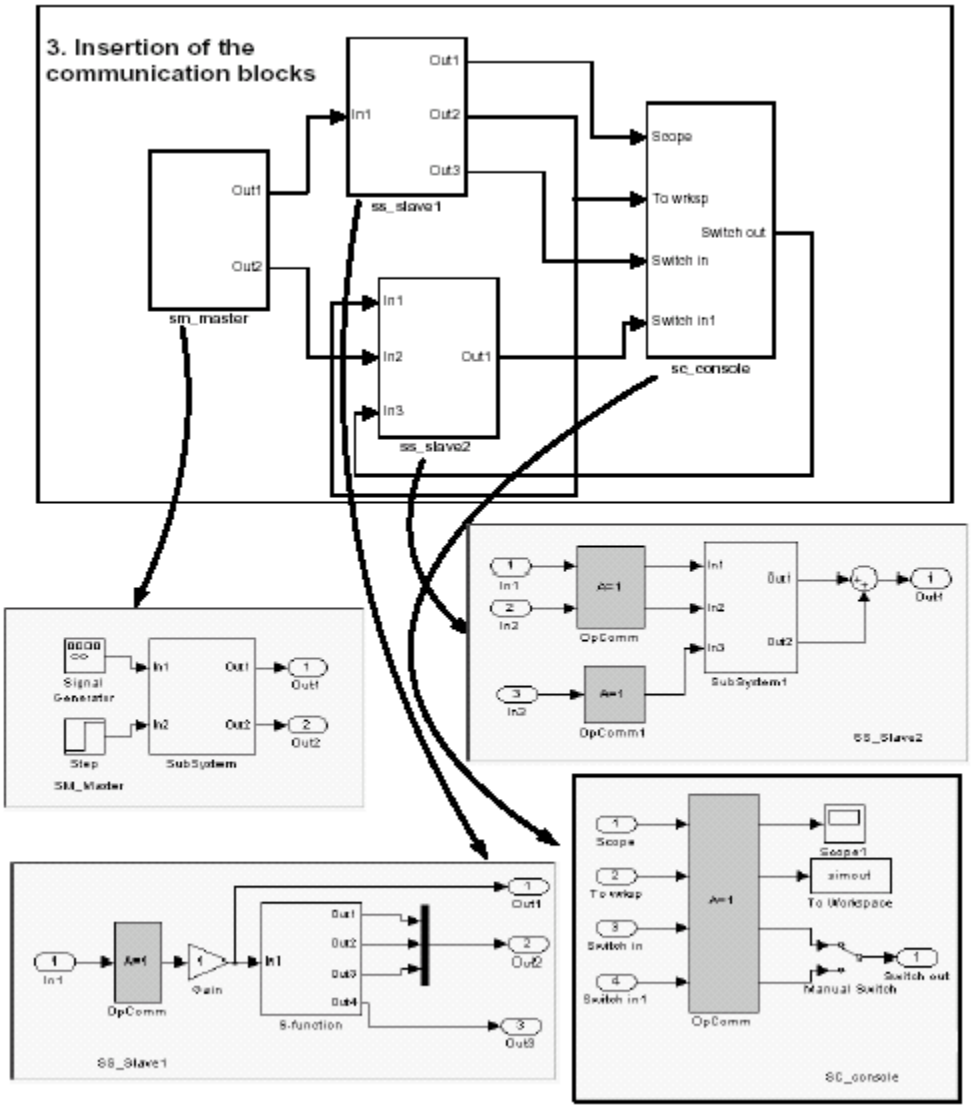
- All blocks must be included in the subsystems: the top-level model must only show the grouped subsystems. Name the subsystems with a prefix indicating the function of that subsystem: *SC_* for Console and *SM_* for Master subsystem respectively.

- A model must have one *Console Subsystem* (*SC_*) and one Master calculation subsystem (*SM_*).

- None of the two types of subsystems (*SC_*, *SM_*) may include any other type of subsystem

2.1.5 Inserting OpComm Communication Blocks

This section explains how and when to insert OpComm blocks into your block diagram, and discusses OpComm parameters that are specific to Simulink users, and those specific to SystemBuild users. You can find the *OpComm* block in the RT-Lab section of the Simulink Library Browser. The following figure demonstrates these points with an example of shaded OpComm icon inserted in a Simulink model.



In the above diagram, you may find there is "new" subsystem with the name "SS_XXX". The subsystems with "SS_" in their names are *Slave Subsystem*. For the Solo version of RT-Lab, we do not have to worry about them. If you want to get more detailed information about them, please go to the online help system or check the related description about them in Opal-RT lab's website at

Http://www.Opal-RT.com.

2.1.5.1 OpComm blocks

Once the model is grouped into Console and computation subsystems, special blocks called OpComm must be inserted into the subsystems. These are simple feed-through blocks that intercept all incoming signals before sending them to computation blocks within a given subsystem. OpComm blocks serve three purposes:

1. When a simulation model runs in the RT-LAB environment, all connections between the main subsystems (*SC_*, *SM_*) are replaced by hardware communication links.

2. OpComm blocks provide information to RT-LAB concerning the type and size of the signals being sent from one subsystem to another.

3. OpComm blocks inserted into the *Console Subsystem* allow you to select the data acquisition group you want to use to acquire data from the model, and to specify acquisition parameters.

2.1.5.2 Rules for inserting OpComm blocks

1. When inserting OpComm blocks into the Console (*SC_*) subsystem:

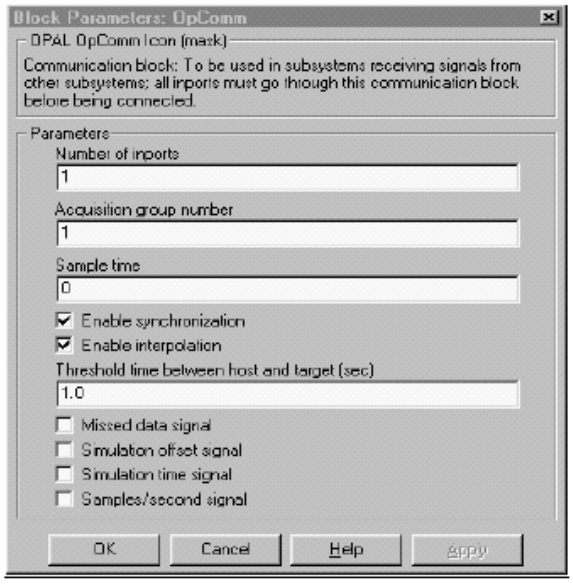
   There must be a maximum of one OpComm block for each Acquisition Group. The Acquisition Group number is specified in the OpComm mask.

2. When inserting OpComm blocks into a Master (*SM_*) or Slave (*SS_*) subsystem:

   There are two kinds of communication associated with Target Node (real-time and non-real-time) but these cannot be sent through the same OpComm block. There must be a maximum of one OpComm block for all real-time communication (communication between Target Nodes), and a maximum of one OpComm block for all non-real-time communication (communication between the Console and a Target Node), for a total maximum of two OpComm blocks in any Master (*SM_*) or Slave (*SS_*) subsystem.

2.1.5.3 OpComm parameters

Double clicking on the OpComm block, you can bring up the following dialog box, which enables you to edit the parameters of OpComm block.



Button / Field	Function
Number of Inports	Indicates the number of subsystem input ports (called inports in Simulink) that are to be intercepted by an OpComm block. The inports to a subsystem are simply the signals coming from another subsystem. The block dynamically adjusts the number of its input and output pins according to this parameter.
Acquisition Group	This parameter, represented by an A on the icon, is relevant only when the OpComm block is inserted into the Console (*SC_*) subsystem. It has no effect when the OpComm block is inserted into a Master (*SM_*) or Slave (*SS_*) subsystem. This parameter specifies the acquisition group number (1, 2, 3, ... 25) for all signals that pass through the OpComm block.
Sample Time	This parameter is only used for multirate simulation.

2.1.5.4 OpComm acquisition parameters

The RT-LAB OpComm icon mask offers the following input and output signals.

Button / Field	Function
OpComm acquisition parameters	Enables the synchronization algorithm to keep the right signal's shape. (Input)
Enable interpolation	Enables the linear interpolation between two values during data loss. (Input)
Threshold	Defines the threshold difference between the simulation Command Station time and the simulation target time. (Input)
Missed data	Number of data lost from the simulation platform. Due to network congestion or CPU use, the Console (*SC_*) is unable to refresh the display with all the data coming from the simulation target platform. In this case, a reception algorithm reacts to the situation to keep the Console synchronized with the simulation target. (Output)
Simulation time	Time elapsed since the beginning of the simulation's execution on the target platform. (Output)
Simulation offset	Time elapsed since the Console started acquiring data from the simulation target platform. (Output)
Samples / second	Number of samples displayed by the Console between two missed data packets. Can also be used to infer average data displayed. (Output)

3 Running your Simulation

After distributing the model and generating its associated C code as described in the previous chapters, next in the model implementation process is compilation of the model, and finally its simulation. This chapter explains the procedures involved in compiling, simulating offline, loading your model, executing its simulation, and interacting with the model during simulation.

3.1 Compiling

Compiling your distributed model is a fully automated process, initiated by clicking on the Compile button on the Main Control Panel. A dialog box will present four check boxes to be completed before you proceed. Right-click Compile to reset the parameters.

When you click Compile, RT-LAB will:

- Separate the Model into the SC_, SM_, and SS_ subsystems you defined.

  The now-grouped block diagram will be split into smaller diagrams, each associated with a different processor. There will be one diagram generated for each *SC_*, *SM_*, or *SS_* subsystem.

- Automatically generate code on the Master and Slave computers.

  The Simulink sub-models are coded into C language by the MATLAB Real-Time Workshop (RTW) module. RTW also creates a makefile according to a specified template. Since real-time models are executed under the QNX environment, the template used is one designed for compiling under the QNX operating system. During execution of the model's real-time simulation, the *SC_* model can interact with the simulation, since C code is not generated for the Console.

- Automatically compile the C code

  The sub-model files, now coded in C, are then compiled within the QNX environment. Files required for compiling are transferred by Ethernet link from your NT workstation to a workstation operating under QNX. This QNX station compiles all C files for the sub-models, to generate files ready for execution on the Target Node.

- Automatically apply panel settings

  After the code is generated and the model compiled; the user must set execution options before running the simulation. These options, discussed in Chapters 6 through 10, deal with assigning the model to physical CPUs in the network, and setting the configuration parameters through the Configuration panel.

3.2 Simulating offline

Models can be simulated offline, which is within their Simulink or SystemBuild environments, even after subsystems have been defined and communication blocks inserted. This option is particularly useful for isolating various problems that may exist within the model. During real-time execution, problems can arise at many levels in the system, making it hard to determine the exact source of the problem. By running the grouped model's simulation off-line, you can investigate any results associated with delays caused by parallel communication, or by the "sample and hold" of I/O communications, without having to worry about problems related to the peripheral hardware (communication boards, memory allocation, bad connections, etc.). Additionally, those using RT-LAB in a HIL context can modify and check the command system offline, which reduces the risk of damage to the hardware.

To start simulation execution

1. Start the Main Control Panel by double-clicking on the *MainControl.exe* icon on your desktop.

2. Select the Simulink file for the grouped model (the .mdl file) by clicking on the *Open Model* button.

3. Select the target platform, and then click on the *Compile* button in the Main Control Panel. This automatically starts the following processes:

    - Separation of the grouped model into subsystems;
    - C code generation for each subsystem;
    - Code compilation for the various subsystems for real-time execution.

    At the end of this step, an executable file is generated for each subsystem of the model. Each file is executed by a Target Node as previously assigned.

4. Assign the generated executables to the real-time cluster's physical nodes. Click on the *Assign nodes* button to bring up the list of executables. These constitute the distributed model, and the list of available computing nodes. Each executable can then be assigned to a node in the system.

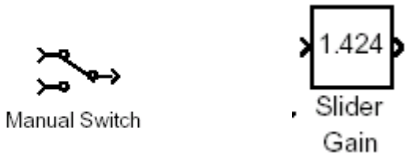
5. Load the executable files by clicking on the *Load* button; if the *Load/Reset Console Also* option is checked, the MATLAB program starts automatically, and the *Console Subsystem* file is displayed on screen. The RT-LAB display window will appear for the various system nodes, displaying simulation diagnostics between the nodes.

6. Click on Execute to start the system.

3.3 Interacting with the model during execution

You can interact with the model and change its parameters during real-time execution via the *Console Subsystem*, located on the Command Station. This feature is useful for adjusting parameters that control model performance: gains, time constants, various limits, etc.

Two typical parameter modification blocks, which can be used to interact with the simulation from the Console, are the *Manual Switch* and the *Slider Gain*:



4 Multifunction I/O Device NI PCI-6036E

The Multifunction I/O device NI PCI-6036E is installed in the target node Iron. It features 16 channels (eight differential) of 16-bit analog input, two channels of 12-bit analog output, a 68-pin connector, and eight lines of digital I/O. You can use it to acquire external data.

4.1 Hardware Overview

This chapter presents an overview of the hardware functions on NI 6036E device.

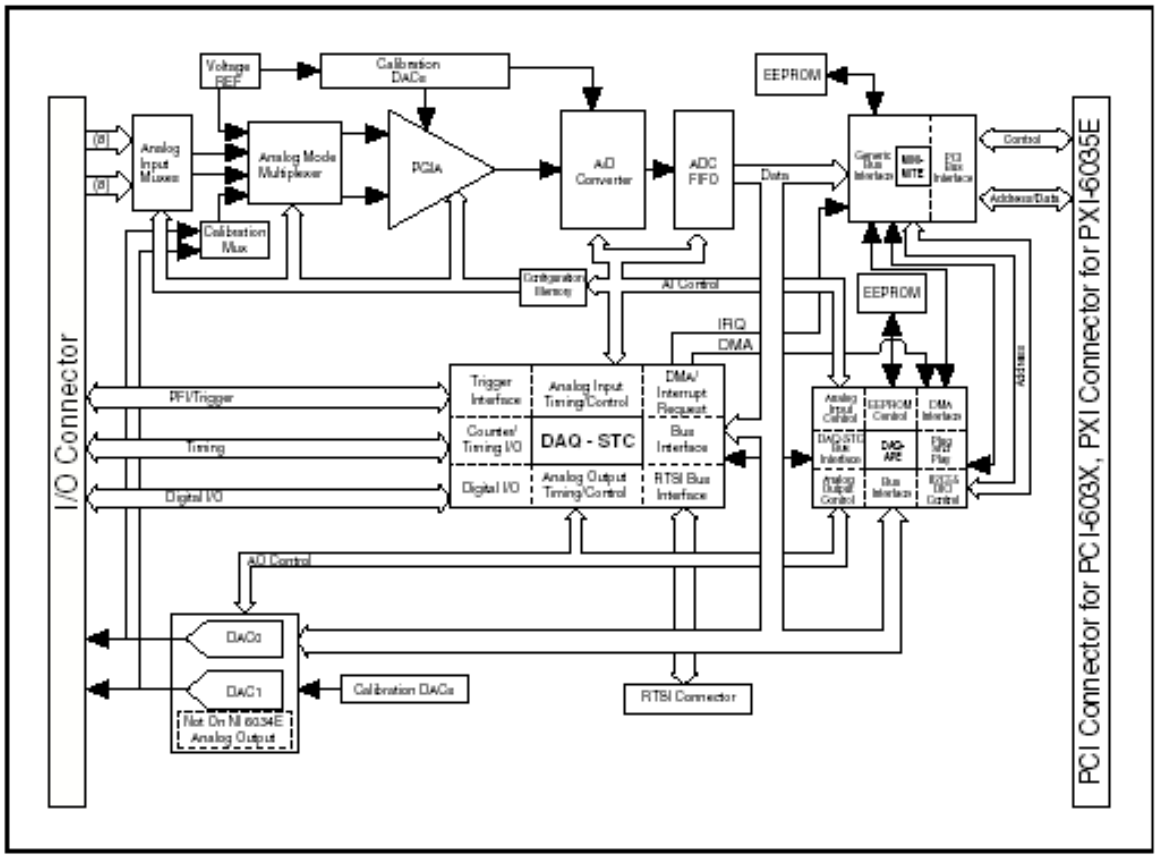


Figure of NI 6036E Block Diagram (From the Official NI User's Manual)

4.1.1 Analog Input

The analog input (AI) section of the NI 6036E device is software configurable. The following sections describe in detail each of the analog input settings.

4.1.1.1 Input Mode

The NI 6036E device has three different input modes— *nonreferenced single-ended (NRSE) input*, *referenced single-ended (RSE) input*, and *differential (DIFF) input*. The single-ended input configurations provide up to 16 channels. The DIFF input configuration provides up to eight channels. Input modes are programmed on a per-channel basis for multimode scanning. For example, you can configure the circuitry to scan 12 channels—four differentially-configured channels and eight single-ended channels.

The following table describes the three input configurations.

Configuration	Description
Diff	A channel configured in DIFF mode uses two analog input lines. One line connects to the positive input of the programmable gain instrumentation amplifier (PGIA) on the device, and the other connects to the negative input of the PGIA.
RSE	A channel configured in RSE mode uses one analog input line, which connects to the positive input of the PGIA. The negative input of the PGIA is internally tied to analog input ground (AIGND).
NRSE	A channel configured in NRSE mode uses one analog input line, which connects to the positive input of the PGIA. The negative input of the PGIA connects to analog input sense (AISENSE).

4.1.1.2 Input Range

The NI 6036E device has a bipolar input range that changes with the programmed gain. Each channel may be programmed with a unique gain of 0.5, 1.0, 10, or 100 to maximize the 16-bit analog-to-digital converter (ADC) resolution. With the proper gain setting, you can use the full resolution of the ADC to measure the input signal.

The following table shows the input range and precision according to the gain used.

Gain	Input Range	Precision
0.5	-10 to +10 V	305.2 μV
1.0	-5 to 5 V	152.6
10	-500 to 5000 mV	15.3
100	-50 to 50 mV	1.53

4.1.2 Analog Output

NI 6036E device supplies two channels of 16-bit analog output voltage at the I/O connector. Each device has a fixed bipolar output range of ±10 V. Data written to the digital-to-analog converter (DAC) is interpreted as two's complement.

For the Description of Digital I/O and other device function, Please refer to the user's manual of NI PCI-6036E.

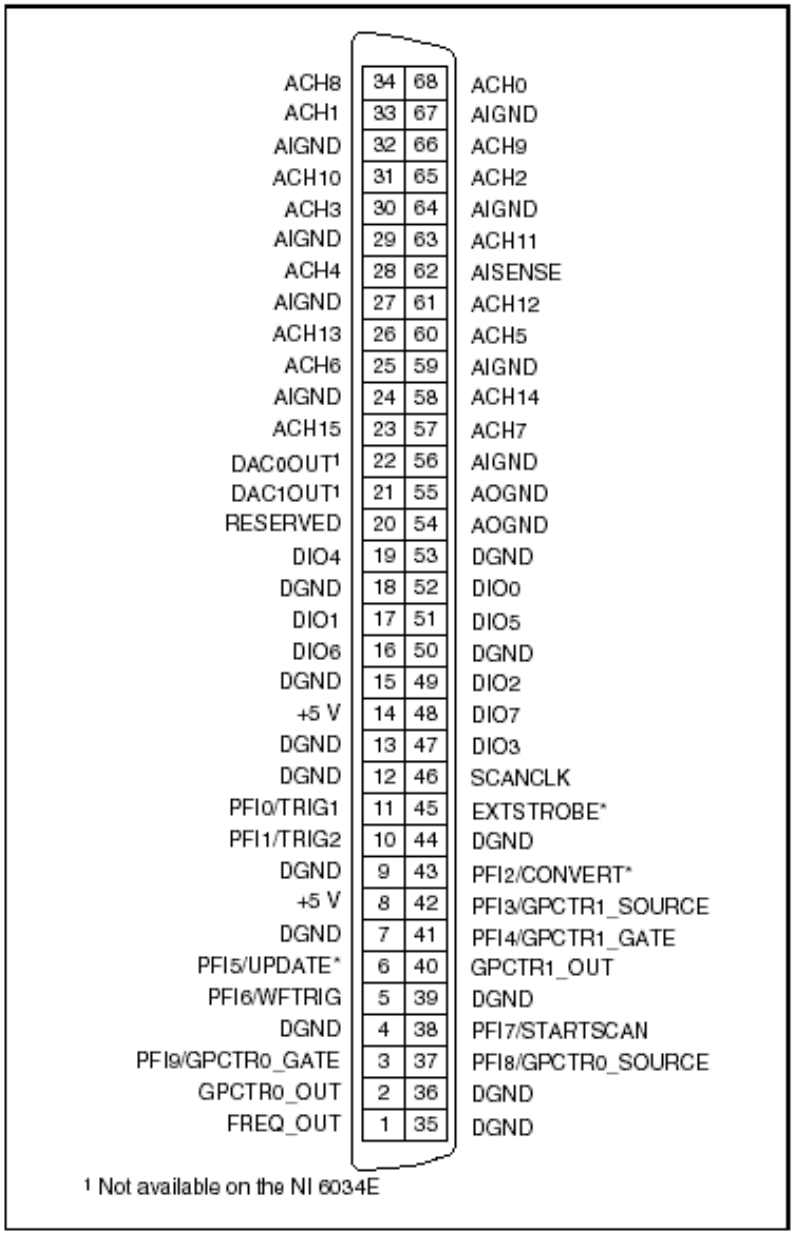
4.2 Connecting you signal

This chapter describes how to make input signal connections to your NI 6036E device using the I/O connector.

4.2.1 I/O Connector

The figure in the next page shows the pin assignments for the 68-pin I/O connector.

Caution: Connections that exceed any of the maximum ratings of input or output signals on the NI 6036E device can damage the device and the computer!



Pin assignments for the 68-pin I/O connector
(From the Official NI User's Manual)

Signal Descriptions for I/O Connector Pins

Signal Name	Reference	Direction	Description
AIGND	—	—	Analog Input Ground—These pins are the reference point for single-ended measurements in RSE configuration and the bias current return point for differential measurements. All three ground references—AIGND, AOGND, and DGND—are connected together on your device.
ACH<0..15>	AIGND	INPUT	Analog Input Channels 0 through 15—Each channel pair, ACH<i, i+8> (i = 0..7), can be configured as either one differential input or two single-ended inputs.
AISENSE	AIGND	INPUT	Analog Input Sense—This pin serves as the reference node for any of channels ACH<0..15> in NRSE configuration.
DAC0OUT1	AOGND	OUTPUT	Analog Channel 0 Output—This pin supplies the voltage output of analog output channel 0.
DAC1OUT1	AOGND	OUTPUT	Analog Channel 1 Output—This pin supplies the voltage output of analog output channel 1.
AOGND	—	—	Analog Output Ground—The analog output voltages are referenced to this node. All three ground references— AIGND, AOGND, and DGND—are connected together on your device.
DGND	—		Digital Ground—This pin supplies the reference for the digital signals at the I/O connector as well as the +5 VDC supply. All three ground references—AIGND, AOGND, and DGND—are connected together on your device.
DIO<0..7>	DGND	INPUT or OUTPUT	Digital I/O signals—DIO6 and 7 can control the up/down signal of general-purpose counters 0 and 1, respectively.
+5 V	DGND	OUTPUT	+5 VDC Source—These pins are fused for up to 1 A of +5 V supply. The fuse is self-resetting.
SCANCLK	DGND	OUTPUT	Scan Clock—This pin pulses once for each A/D conversion in scanning mode when enabled. The low-to-high edge indicates when the input signal can be removed from the input or switched to another signal.
EXTSTROBE	DGND	OUTPUT	External Strobe—This output can be toggled under software control to latch signals or trigger events on external devices.

Signal Descriptions for I/O Connector Pins (Continued)

Signal Name	Reference	Direction	Description
PFI0/TRIG1	DGND	INPUT OUTPUT	PFI0/Trigger 1—As an input, this signal is one of the Programmable Function Inputs (PFIs). PFI signals are explained in the Connecting Timing Signals section later in this chapter. As an output, this signal is the TRIG1 (AI Start Trigger) signal. In posttrigger data acquisition sequences, a low-to-high transition indicates the initiation of the acquisition sequence. In pretrigger applications, a low-to-high transition indicates the initiation of the pretrigger conversions.
PFI1/TRIG2	DGND	INPUT OUTPUT	PFI1/Trigger 2—As an input, this signal is one of the PFIs. As an output, this signal is the TRIG2 (AI Stop Trigger) signal. In pretrigger applications, a low-to-high transition indicates the initiation of the posttrigger conversions. TRIG2 is not used in posttrigger applications.
PFI2/CONVERT	DGND	INPUT OUTPUT	PFI2/Convert—As an input, this signal is one of the PFIs. As an output, this signal is the CONVERT (AI Convert) signal. A high-to-low edge on CONVERT indicates that an A/D conversion is occurring.
PFI3/GPCTR1_SOURCE	DGND	INPUT OUTPUT	PFI3/Counter 1 Source—As an input, this signal is one of the PFIs. As an output, this signal is theGPCTR1_SOURCE signal. This signal reflects the actual source connected to the general-purpose counter 1.
PFI4/GPCTR1_GATE	DGND	INPUT OUTPUT	PFI4/Counter 1 Gate—As an input, this signal is one of the PFIs. As an output, this signal is the GPCTR1_GATE signal. This signal reflects the actual gate signal connected to the general-purpose counter 1.1.
GPCTR1_OUT	DGND	OUTPUT	Counter 1 Output—This output is from the general-purpose counter 1 output.
PFI5/UPDATE	DGND	INPUT OUTPUT	PFI5/Update—As an input, this signal is one of the PFIs. As an output, this signal is the UPDATE (AO Update) signal. A high-to-low edge on UPDATE indicates that the analog output primary group is being updated for the NI E and NI 6036E.
PFI6/WFTRIG	DGND	INPUT OUTPUT	PFI6/Waveform Trigger—As an input, this signal is one of the PFIs. As an output, this signal is the WFTRIG (AO Start Trigger) signal. In timed analog output sequences, a low-to-high transition indicates the initiation of the waveform generation.
PFI7/STARTSCAN	DGND	INPUT OUTPUT	PFI7/Start of Scan—As an input, this signal is one of the PFIs. As an output, this signal is the STARTSCAN (AI Scan Start) signal. This pin pulses once at the start of each analog input scan in the interval scan. A low-to-high transition indicates the start of the scan.

Signal Descriptions for I/O Connector Pins (Continued)

Signal Name	Reference	Direction	Description
PFI8/GPCTR 0_SOURCE	DGND	INPUT OUTPUT	PFI8/Counter 0 Source—As an input, this signal is one of the PFIs. As an output, this signal is the GPCTR0_SOURCE signal. This signal reflects the actual source connected to the general-purpose counter 0.
PFI9/GPCTR 0_GATE	DGND	INPUT OUTPUT	PFI9/Counter 0 Gate—As an input, this signal is one of the PFIs. As an output, this signal is the GPCTR0_GATE signal. This signal reflects the actual gate signal connected to the general-purpose counter 0.
GPCTR0_OU T	DGND	OUTPUT	Counter 0 Output—This output is from the general-purpose counter 0 output.
FREQ_OUT	DGND	OUTPUT	Frequency Output—This output is from the frequency generator output.

4.2.2 Analog Input Signal Overview

The analog input signals for the NI 6036E device are ACH<0..15>, ASENSE, and AIGND. Connection of these analog input signals to your device depends on the type of input signal source and the configuration of the analog input channels you are using. This section provides an overview of the different types of signal sources and analog input configuration modes.

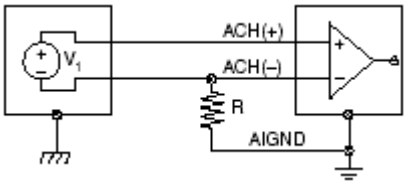
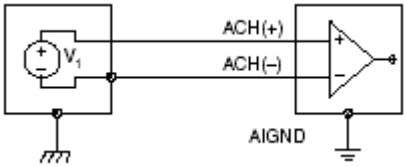
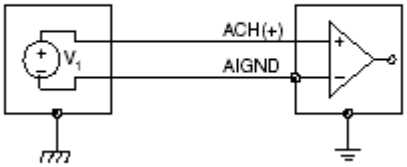
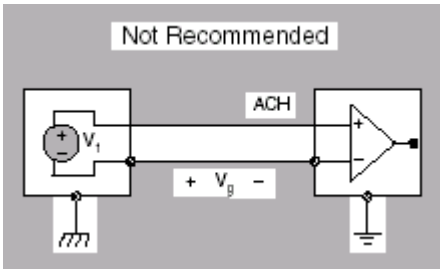
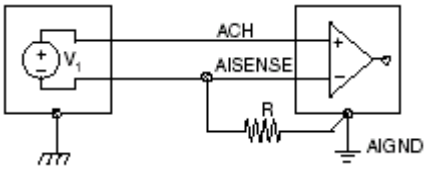
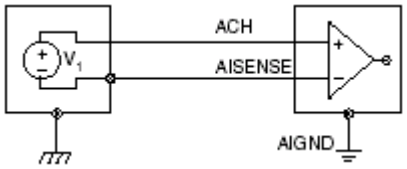
4.2.2.1 Analog Input Modes

You can configure your device for one of three input modes: nonreferenced, single ended (NRSE), referenced single ended (RSE), and differential (DIFF).

4.2.2.2 Connecting Your Analog Input Signals

The figure in the next page summarizes the recommended input configuration for single-ended and differential connection modes.

Summary of Analog Input Connections

INPUT	INPUT	
	Floating Signal Source (Not Connected to Building Ground)	Grounded Signal Source
INPUT	Examples • Underground Thermocouples • Signal Conditioning with Isolated Outputs • Battery Devices Example	• Plug-in Instruments with Nonisolated Outputs
Differential (DIFF)		
Single-Ended — Ground Referenced (RSE)		
Single-Ended — Nonreferenced (NRSE)		

5 RT-Lab Blocks

RT-LAB comes with a library of blocks that can be added to your model. These allow access to different types of I/O modules and other functions. This chapter will discuss the need, use, and parameters of the I/O blocks.

RT-LAB block libraries can be found in the Simulink library browser.

To add blocks to your model, and set their parameters:

1. Click on the RT-LAB block that corresponds to the desired function of the I/O card you are using.
2. Drag and drop the RT-LAB block into your simulation model.
3. Double-click the RT-LAB block to change its parameters.

This will open a parameters control box for the block in question.

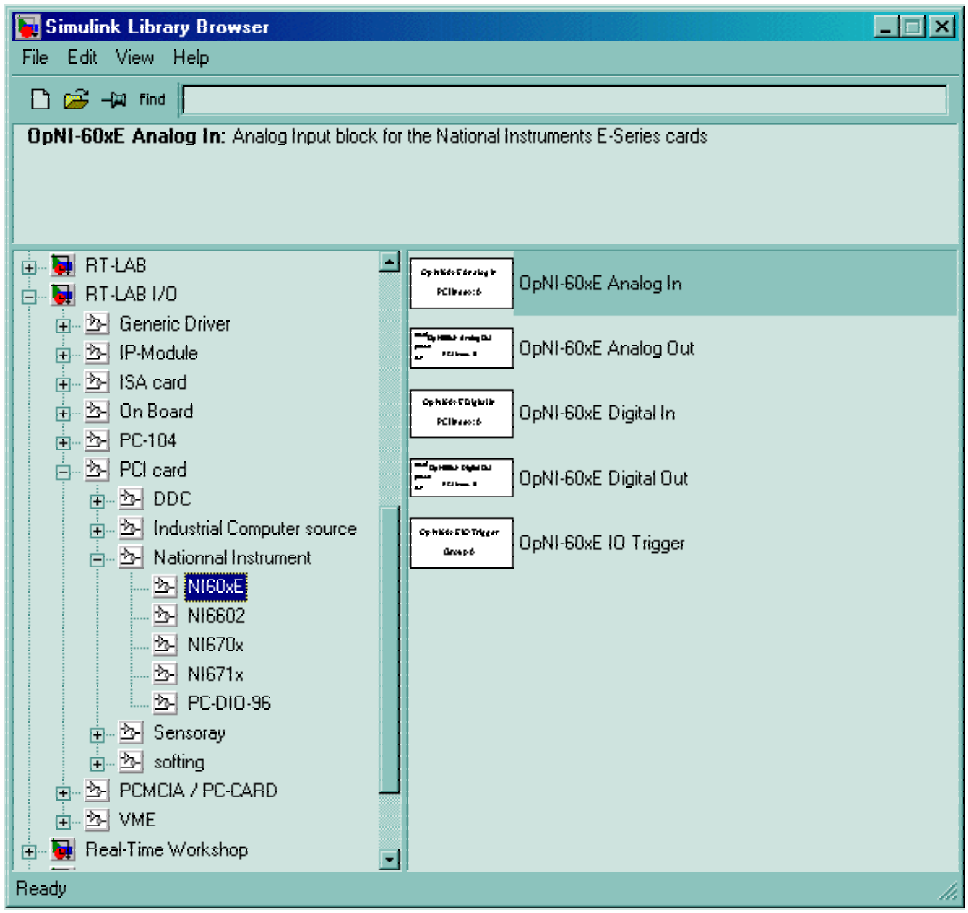
I/O modules allow users to control external equipment. These modules may be digital to analog (D/A) converters, analog to digital (A/D) converters, input/output (I/O) binary ports, and quadrature decoders, among others.

To find the RT-LAB I/O block library:

In Simulink, the blocks are located under RT-LAB I/O in the Simulink library browser: Type opal-lib in the Matlab command window, then press <Return>.

5.1 RT-LAB I/O Module blocks

The block library includes blocks for every available type of I/O module. Since we are using the NI PCI 6036E. You can easily find it in the Simulink Library Browser. The following Figure shows you the position of it.

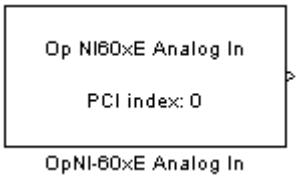


The I/O module blocks discussed are:
• OpAnalogIn
• OpAnalogOut

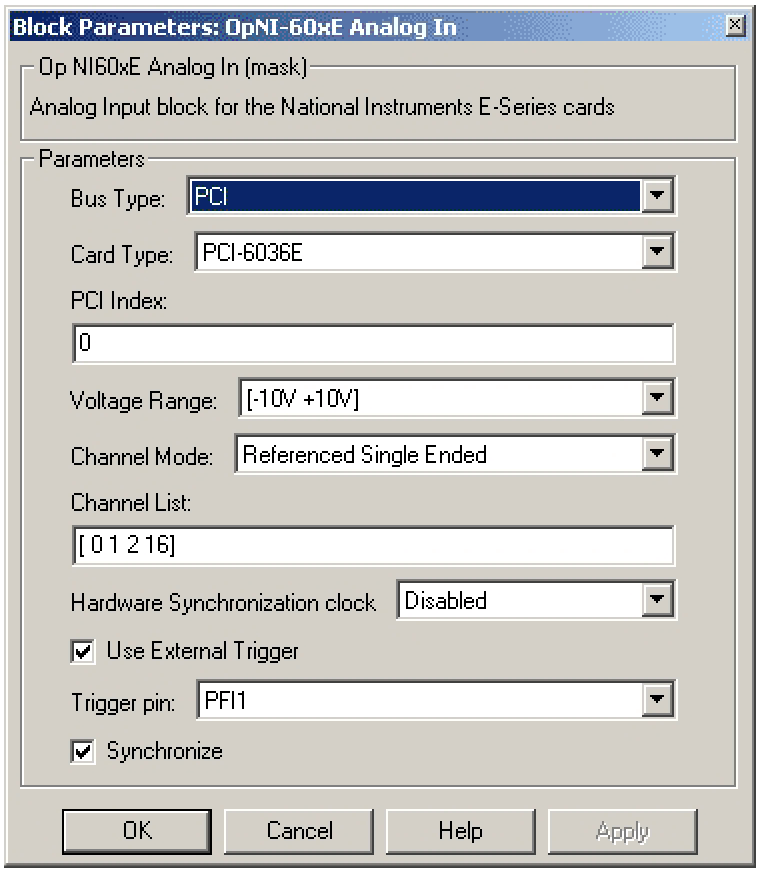
Do be careful when you are using PCI-6036E to acquire data. Exceeding any of the maximum signal ratings on the NI6036E device can create a shock or fire hazard, or can damage any or all of the boards connected to the chassis, the computer, and the NI6036E device.

5.1.1 OpAnalogIn

Block



Mask



Description

This block monitors the analog input channels. Each channel has individually configurable mode (referenced sinlge-ended, non-referenced single-ended and differential) and voltage range (Please refer to Chapter 4 *Multifunction I/O Device NI PCI-6036E* for more detailed information). One Op Ni60xE Analog In block supports only one mode and one voltage range for a given list of channel numbers. If the input signals require the use of several modes and voltage ranges, multiple OpNi60xE Analog In blocks specifying different lists of channels must be used in the RT-LAB model. This block supports a list of and PCI compatible cards including our PCI-6036E. The block

automatically updates the list of available voltage ranges depending on the card model selected by the user. For card specific channel number and resolution, please refer to NI documentation. This block can also be used for hardware synchronization. Trigger source may either come from an onboard counter or from an external signal. XHP mode is supported only when ' Hardware Synchronization clock' and 'Use External Trigger' are both disabled. For more detailed information about synchronization, please refer to the official manual of PCI-6036E from National Instruments.

Parameters

Bus Type: Select the bus type (PCI or PXI) of the target node. (PCI in our case)
Card Type: Select the card model. (PCI-6036E in our case)
PCI index: Enter the PCI index of the card on the PCI bus. (0 in our case)
Voltage range: Select the voltage range that best fits the expected input signal range. You have 4 options. -10V ~ +10V, -5V ~ +5V, -500mV ~ +500mV, -50mV ~ +50mV.
Channel Mode: Select the channel polarity mode in this list. Make sure that the hardware connections of the input signals are consistent with this selection.
Channel List: Enter the list of channels that will be monitored by this block. The channels can be entered in any order. If some channels require a different voltage range or channel mode than those selected above, another block must be used. Make sure that a given channel does not appears in the channel of only one blocks.
Hardware Synchronization clock: Several hardware synchronization options are available.
- *Disabled*: Do not use this board for hardware synchronization.
- *External Clock*: The model will synchronize on an external signal fed into the trigger pin. The actual time step of the model will be determined by the period of the external signal. The model must be running in 'Hardware synchronize' mode. Note that the 'Time Factor' parameter of the MainControl panel has no meaning when the model is running in this mode. Use the *Trigger Pin* parameter to specify the board input to use as a trigger source. Analog inputs sampling also triggered by the signal.
- *Internal Clock*: An internal timer, programmed at the given model fixed step-size will be used to synchronize the model. The model must be running in 'Hardware synchronized' mode. Analog inputs sampling also occurs when the internal timer triggers the model.
Use External Trigger: Select this option to allow an external signal to start the A/D conversion. The external signal should be fed into one of the available PFI pins listed in the 'Trigger pin' . All blocks belonging to the same IO board must have this parameter set to the same value. This parameter must be checked if hardware synchronization clock is set to External clock. Also, no external trigger source may be used when hardware synchronization clock is set to Internal clock.

Trigger Pin: Select the PFI pin that will be connected to the external trigger signal. The external GND should be connected to one of the DGND pins.

Inputs

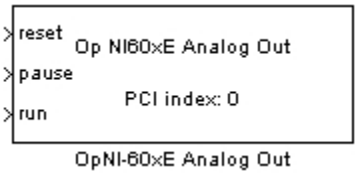
This block has no inputs.

Outputs

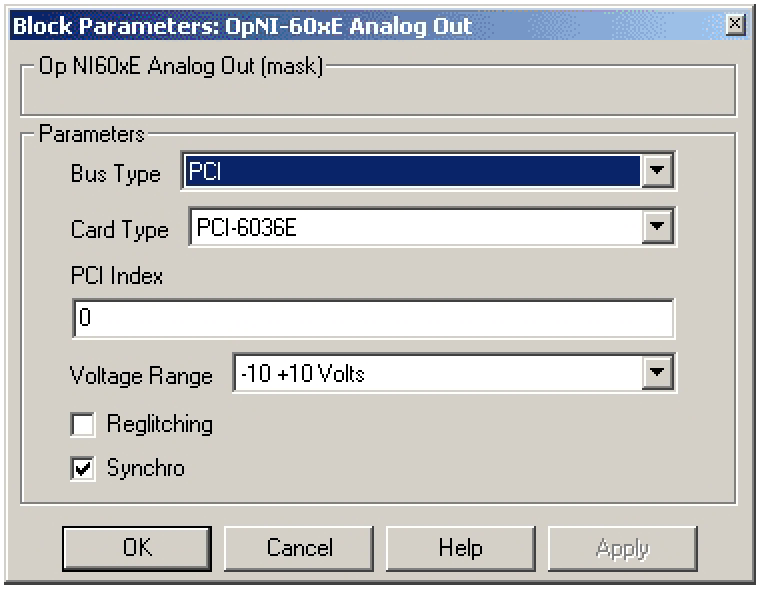
At each calculation step of the model, the block returns the current values of the input signals connected to the card.
Only the values of the channels specified in the channel list parameter are returned and the outputs are in the order of the 'channel list' parameter. The width of the output port of the block must be equal to the number of entries in the channel list parameter. Use a demux to connect individual outputs to the rest of the model.
The values are returned in Volts.

5.1.2 OpAnalogOut

Block



Mask



Description

This block monitors the 2 analog output channels. It automatically updates the list of available voltage ranges depending on the card number selected by the user.

Parameters

Bus Type: Select the bus type (PCI or PXI) of the target node in this list. (PCI in our case)
Card Type: Select the card model in this list. (PCI-6036E in our case)
PCI index: Enter the PCI index of the card on the PCI bus. (0 in our case)
Voltage range: Select in this list the voltage range that best fits the expected output signal range.
Reglitching: Select this option to enable the reglitch circuit. This circuit uniformizes the glitches that occur when output lines are updated with new values. For more detailed information, please refer to National Instruments' *PCI E Series User Manual*

Inputs

At each calculation step of the model, the block sets the analog output channels to the voltage values provided by the block inputs. The input values are assumed to be in Volts.
When an external voltage is used to define the voltage range, the input values calculated by the model (V_{model}) must be scaled to the external voltage reference value (V_{ExtRef}) before being sent to the block. For this reason, it is recommended to connect the external voltage reference signal to an analog input channel so that the model knows its exact value and can use it for scaling the analog outputs. The actual voltage values submitted to the block should be equal to:

$V_{model} / V_{ExtRef} * 10.0V$

Three sets of input values must be specified. Each of the three input vectors must specify the voltage values to be set to both analog output channels.
Reset values: Values that will be written to the analog out channels when the reset button is hit.
Pause values: Values that will be written to the analog out channels when the model is in pause mode, that is at initialization or when the pause button is hit.
Run values: Values that will be written to the analog out channels at each time step.
Note that to ensure that the pause and reset values be properly written out, it is recommended to set these values in the model subsystem that contains the Op-Ni60xE Analog Out block, and not to provide them from the console subsystem.

Connecting a simple constant to the Pause and Reset inputs will not make this value apply to all channels. This value will apply only to the first channel, and all other channels will take the default value (0). Use a multiplexing block to define Reset and Pause values for all channels.

5 Other Useful resources

- Opal-RT Technologies, Inc's website

  http://www.Opal-Rt.com

- RT-Lab Knowledge base

  http://support.opal-rt.com

- RT-LAB general block library

  http://support.opal-rt.com/common/docs/html/api_index.html

- QNX Official website

  http://www.qnx.com

- National Instruments Official Website

  http://www.ni.com

Appendix A: Booting into QNX

QNX has been installed on Iron. To boot into QNX, simply restart the computer and Iron will be booted into QNX automatically. You can Login QNX with user name: `ialabuser` and password: `ialabuser`. (These fields are **case sensitive**.)

To go back to Windows, reboot Iron and type **1** when the following message is shown in screen.

```
Press F1-F4 to select drive or select partition:
```

Appendix B: Getting Started with QNX

Getting started in QNX is easy. Here are simple steps to work on a QNX machine.

Logging in

- Logging in: To log in you have 2 prompts. Both fields are **case sensitive**. The first prompt will be your username; the second prompt will be your password.
- After you log into QNX's graphic interface, click the "**terminal**" button to open a text window for typing QNX commands

Directory Commands:

- Tell you what directory you are in: **pwd**
  To find out what directory you are in, use **pwd**, which stands for print working directory.

- Changing directories: **cd** ***directory***
  To change your directory simply use **cd** ***directory-name***. In QNX, if you are using a full path, use the forward slash. For example, a user named "ialabuser" may have his home directory in /home/ialabuser. Ialabuser can get to his default home directory by simply typing **cd**. If he wants to be in /tmp, he would type **cd /tmp**
  - ***/ means "top directory"***
  - ***. means "this directory"***
  - ***.. means "the directory above this directory" (parent) as in cd ..***

- Making directories: **mkdir** ***directory***
  Use the **mkdir** command to make one or more directories

- Make yourself a working directory for your stuff: ***mkdir /myname***

- Removing directories: **rmdir** ***directory***
  Use the **rmdir** command to remove one or more directories.

File Commands:

- Listing files: ***ls***
  To list files you use the ***ls*** command. This simply lists the files in a directory. There are many variations of this command. That is, you can add options to see hidden files, time stamps of files, and permissions. If you ***ls filename***, you will see is filename exists. If filename is actually a directory, you will be able to see the contents of that directory.

- ***ls -l* is a "long" format listing with more info** Often the first thing one does is do a ***ls*** when they log into a system. This allows them to see what files they are using and are working on.

- Moving files: ***mv***
  You move files with the ***mv*** command. Simply ***mv file newfilename***. You can move a file to another filename or into a directory.

- Copying files: ***cp***
  You copy files with the ***cp*** command. Simply ***cp file newfilename***. You can copy a file to another filename or into a directory.

- Removing files: ***rm***
  You remove files with the ***rm*** command. Simply ***rm file newfilename***. You can remove many files at once. Unlike windows, you can NOT back out of a remove. Be certain that you want to remove a file.

  Attention Please: Do not try to remove the files in the folder *ntuser*. These may be required for the running of RT-Lab system.

- Editing files: ***vi*** or ***pico***
  As each editor of files is vastly different, you will want to learn these separately. Usually, new users will find ***pico*** easiest to learn.

- QNX comes with the ped editor: *ped* filename

Getting Help:

- *QNX has no **man** command - see the html or online reference docs instead.*

Compiling and running programs:

- to compile***:*** **qcc yourprogram.c servotogo.c -o executablefilename**

- sometimes you need to link in a library like **-***lsocket* (no space between -l and socket)

- to run it: **./executablefilename**

Other things:

- use ftp to transfer files to/from other machines
- voyager is the web browser
- ped is the text editor

Appendix C: Some Important Tips

• Always remember to RESET your model after running you simulation with RT-Lab. Or you model may not running properly next time you want to run it. The RT-Lab may even reject running your model for the next time if you forget to reset. You can do this by clicking the RESET button in the main control panel. In case you forget to do so, change your model's name so it can be accepted again**. Changing the name is not recommended since there will be a lot of junk files left in Command Station and Target Node.**

• Group the *OpAnalogIn* or other I/O blocks into the subsystem that will be assigned to the Target Node, which has the I/O board **installed.** Or the *OpAnalogIn* block will not work. This is very important if you want to acquire data from External System.

• You can use the FTP to do the file transfer work between the Command Station and Target Node. To do this,

1, Open the Ms-Dos Terminal Window in Copper
2, Type ftp
3, Type open
4, Type the IP Address of the Iron at the prompt "132.206.72.102"
5, Then you can use the FTP Command to do your FTP work. To get the help for the ftp command, type "`help`". To get help for certain command, type "`help` *`Your command`*" For instance, if you want to get the help for FTP command SEND. Type "`help send`"

There will be more tips added as I update this Manual. Please come back and check this section each time the manual is updated.

• Do be careful when you are using PCI-6036E to acquire data. Exceeding any of the maximum signal ratings on the NI6036E device can create a shock or fire hazard, or can damage any or all of the boards connected to the chassis, the computer, and the NI6036E device.

Good luck and have fun with RT-Lab!